CMPSC265 Data Structures and Algorithms
Homework 8

Due: Sunday, Nov 3rd, 2019, 11:59pm

Learning Objectives

- Binary Search Tree
- Recursion

Deliverables:

- The modified java file *BinarySearchTree.java* for Problem 1 and 2
- The modified java file TwoDTree.java for Problem 3.

Instructions:

- Please first download the Java source file BinarySearchTree.java and TwoDTree.java, modify the codes based upon requirements, and submit the updated source file. Please do NOT change the provided part of the codes, but you may add additional methods if needed. For Problem 3, you also need to download the *input10K.txt* file as input data.
- Please add the required *credit comments* and *comments* to each Java program you submit.
- **Please TRY YOUR BEST before seeking help from any online resources or tutors or other people. If you have asked for any kind of help, you need also submit your own draft before getting the help, and write a report of how you got helped and what you have learned from the help.**

Problem Specifications:

Problem 1: Is a Binary Search Tree or Not (30')

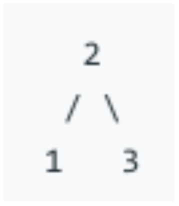
Description:

Please finish the isBST() method in the BinarySearchTree class that will determine whether the tree is a binary search tree or not.

Please remember a valid Binary Search Tree is defined as follows:

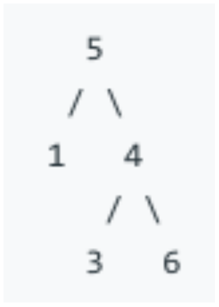
- The left subtree of a node contains only nodes with key less than the node's key.
- The right subtree of a node contains only nodes with key greater than or equal to the node's key
- Both the left and right substrees must also be a binary search tree.

For Example:
The following tree is a Binary Search Tree



```
    2
   / \
  1   3
```

However, the following tree is NOT a binary search tree.



```
    5
   / \
  1   4
     / \
    3   6
```

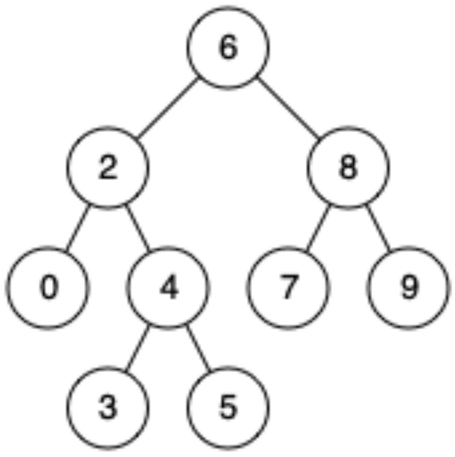
<u>Outputs:</u>

```
$ java BinarySearchTree
The fist tree is a Binary Search Tree
The second tree is NOT a Binary Search Tree.
```

Problem 2: ***Lowest Common Ancestor of a Binary Search Tree (34')***
<u>Description:</u>
Given a binary search tree (BST), find the lowest common ancestor (LCA) of two given nodes in the BST.

According to the definition of LCA on Wikipedia: 'The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (a node can be a descendant of itself')

For example, given the following binary search tree



The LCA of nodes 2 and 8 is 6
The LCA of nodes 4 and 7 is 6
The LCA of nodes 2 and 3 is 2.
The LCA of nodes 7 and 9 is 8.

If any of the two nodes p or q does not exist in the BST, you also need to prompt an error message about it.

Outputs:

```
The LCA of nodes 2 and 8 is 6
The LCA of nodes 0 and 3 is 2
There is no such a node 10 on the tree
There is no such a node 1 on the tree
```

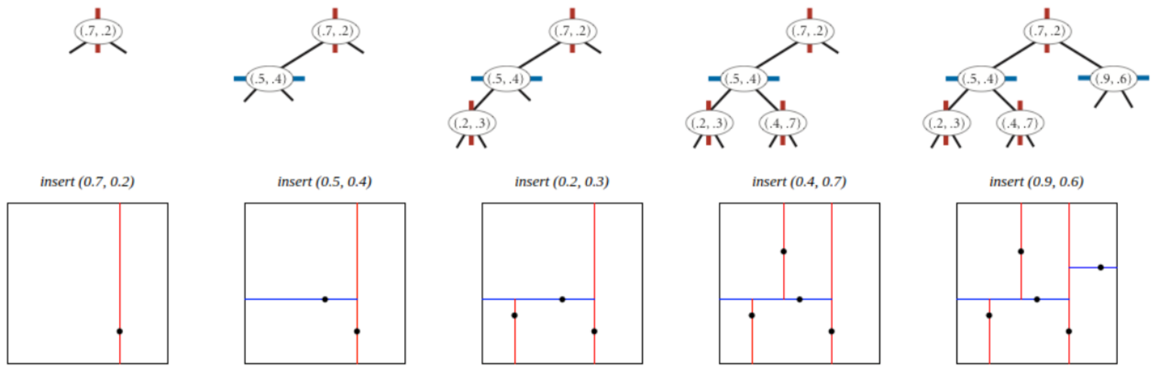
Problem 3: TwoDTree (36')

Description:

Write a data type TwoDTree that represents a 2d-Tree. A 2d-Tree is a generalization of a BST to two-dimensional keys. The idea is to build a BST with points in the nodes, using the x- and y-coordinates of the points as keys in strictly alternating sequence, starting with the x-coordinates.

- *Search and insert*. The algorithms for search and insert are similar to those for BSTs, but at the root we use the x-coordinate (if the point to be inserted has a smaller x-coordinate than the point at the root, go left; otherwise go right); then at the next level, we use the y-

coordinate (if the point to be inserted has a smaller y-coordinate than the point in the node, go left; otherwise go right); then at the next level the x-coordinate, and so forth.
The following figure shows one example of inserting five points into the tree.



Please implement the *search()* and *insert()* method in the TwoDTree class so that the tree can be built and node searched in the above described way.

- *Display* the tree by level-order traversal: the display () method should display all the points on the tree in level-order.
- Please read all the Points data in the file *input10K.txt* and construct the TwoDTree tree for searching and display.

Outputs:

```
$java TwoDTree
The top 10 Points:
(0.40636, 0.6781)
(0.010189, 0.742363)
(0.740024, 0.021714)
(0.147733, 0.203388)
(0.01869, 0.959379)
(0.970874, 0.005952)
(0.537098, 0.43615)
(0.095179, 0.523114)
(0.371858, 0.169457)
(0.014067, 0.805079)
...............

Is point (0.83415, 0.810859) on the tree?:  true
Is point (0.8400, 0.8919) on the tree?: false
```